Tecnológico de Monterrey

Situación Problema: POO 2

Jorge Luis Nájera Espinosa A01424106

Andrea Carolina Figueroa Orihuela A01424250

15 de Junio de 2023

Programación Orientada a Objetos 2

Mónica Larre Bolaños Cacho

Tecnológico de Monterrey

I. Introducción

II. Diagrama de clases UML

A. Imagen del Diagrama

B. Descripción

1. Clases y métodos
2. Líneas de Relación

III. Ejecución

A. Casos de excepción

B. Uso de temas vistos en clase

IV. Conclusión

Tecnológico de Monterrey

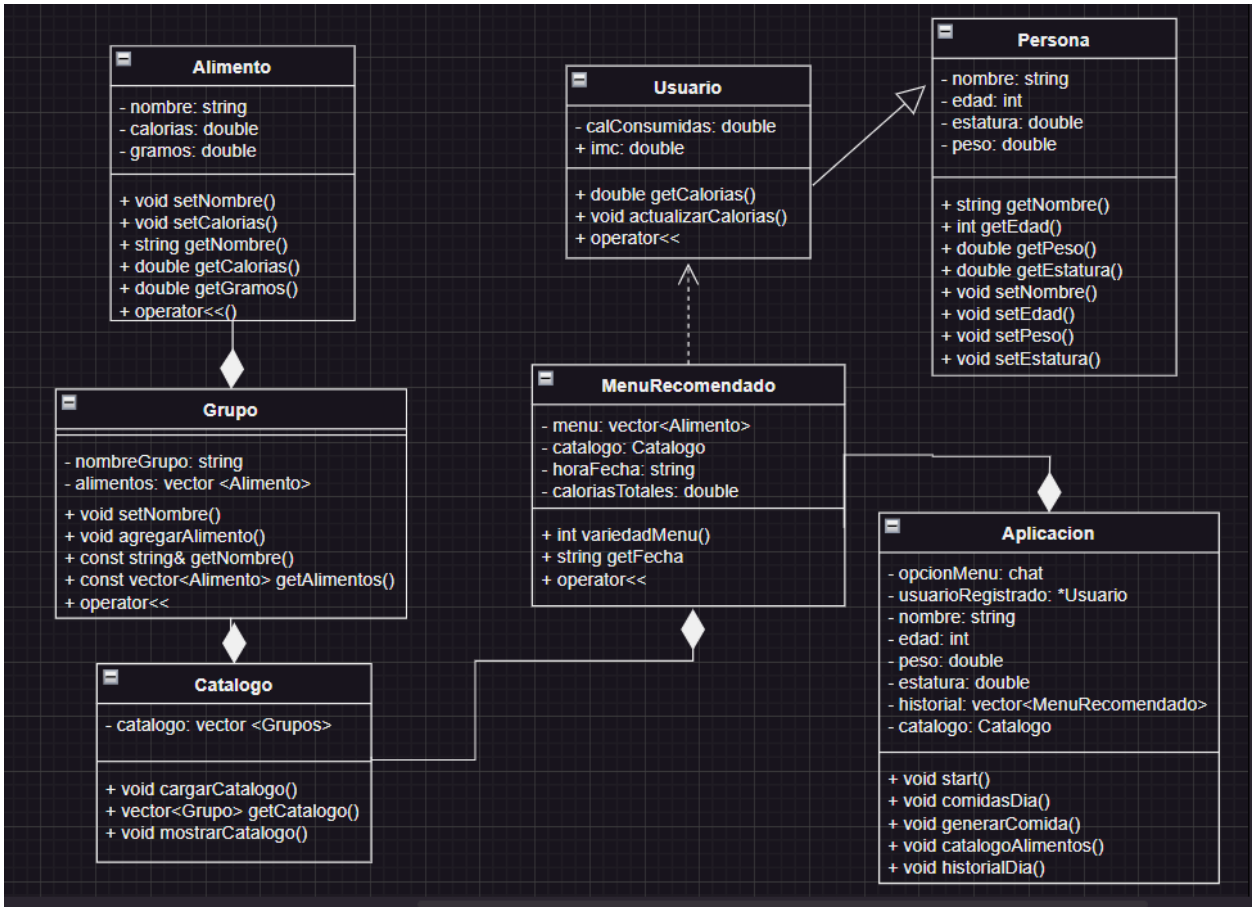
Introducción

El desarrollo de una aplicación de Programación Orientada a Objetos es un proceso el cual a través de este semestre homs ido perfeccionando poco a poco, construyendo sobre trabajos pasados para por fin poder llegar a donde estamos. Empezando desde entender lo que conlleva diseñar un sistema hasta sus mecanismos integrados, en esta situación problema creamos una aplicación de menú inteligente el cual usando los datos e IMC de una persona recomienda platillos o regimenes alimenticios. Sabemos que suena más facil de lo que fue, pero la verdad es que el reto consiste más que nada en implementar los conceptos POO que aprendimos este semestre como lo fue la herencia, el polimorfismo, la sobrecarga o las excepciones. Consiste en darle un sentido a un sistema que se riga bajo normas de seguridad constantes las cuales sirvan como protocolos para que nuestro sistema sea no oslo modular sino adaptable a otros escenarios.

En este caso, construimos dentro de Visual Studio Code un sistema de clases que interactuan con una aplicación ejecutable la cual nos permitirá en base a nuestros datos de salúd como altura, peso y edad; generar desde una comida hasta un menú por día, el cual se adapte de manera saludable y bases científicas a nuestro cuerpo en pos de mejorar nuestra salud a través del régimen alimenticio adecuado. Esto desde un catalogo pre-producido por nosotros en forma de txt el cual contiene todas las comidas a considerar dentro del posible menu con sus porciones y calorías correspondientes. Y a través de métodos que usan todo lo visto en clase; principalmente sobrecarga, manejo de vectores y excepciones; además de temas no tan explorados en clase como los iteradores.

Tecnológico de Monterrey

Diagrama UML



Nuestro sistema de clases, como se verá en nuestra imagen del diagrama final, consiste en siete clases contando la de la app misma (la cual está ubicada en el archivo del main). Cada uno con sus getters y setters, y la clase de Menú Recomendado como el “nexo” central de todo el sistema. Dicha clase tiene una clase usuario y una catalogo. Cabe mencionar que la clase catalogo está conformada por un vector de grupos alimenticios, así como dichos grupos son vectores nombrados de alimentos.

Lo que se encuentra dentro del archivo txt son las comidas con sus tres atributos (nombre, calorías y cantidad recomendada), ordenadas en el orden que se les piensa imprimir; así como ordenadas en grupo pues cada ocho líneas se abre un grupo diferente (seis alimentos por grupo y el nombre del grupo o encabezado).A través de ello podemos entender mejor las líneas de

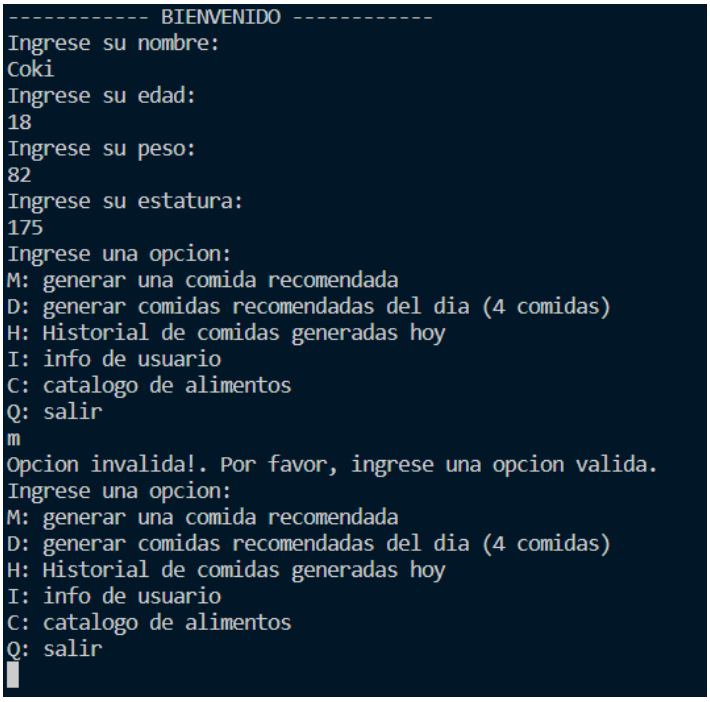
relación del diagrama: como el catalogo es en si una matriz de alimentos, o un vector de vectores de alimentos, se lee el documento como si fueran alimentos directo y se les va emplazando dentro de un vector de grupo (que se cierra y sella con su nombre).

Pasando al otro lado de la tabla, podemos observar a la clase persona, clase base que no interacciona con nada más que la clase usuario. Dicha clase tiene dos atributos, uno dado y otro calculado por sus propios métodos, la cantidad de calorías consumidas y el IMC respectivamente. Pero adquiere sus datos para calcular IMC e imprimir más datos personales heredando su uso de manera publica desde persona. La cual cuenta con cuatro atributos: nombre, peso, edad y estatura

Finalmente, toda esta información es pasada por el usuario al Menu Recomendado por dependencia. Estro porque es la primera clase la que puede tomar decisiones en la segunda, sobre todo la comida que genera, pues este depende de los datos con los que llegue el usuario. Así como ya esta clase estar conformada por el catalogo y el usuario, volviendo al principio y pasando a la aplicación directamente.

La aplicación en si por supuesto también tiene su propia clase como es debido, aunque dicha se encuentre dentro del propio .cpp del main. Esto por cuestiones de conveniencia. Sus atributos incluyen todos los de las clases pasadas pues depende de usuario y su línea de herencia, así como está conformada por Catalogo y su matriz. Sin embargo, al ya ser la clase abstracta con la que interactúa el usuario, dicho usuario tiene que ser uno mismo a través de la sesión. Sus métodos son anidados, pues tiene un método start el cual permite al usuario registrarse con datos nuevos. Así como abrir un menú de uso para el usuario.

Ejecución



Como se mencionado en la parte del diagrama, lo primero que verá el usuario es un cuestionario sobre sus datos de de salud usados para el IMC; además de su nombre para referirlo de alguna forma. En esta parte, cada que iniciemos el programa estaremos creando un usuario nuevo, así que se inicia como un apuntador para no volver a crearlo continuando el programa. También los datos que demos al programa no irán directo a usuario, sino a la clase persona que se los hereda; esto con propósitos de protección y administración. Seguido a ello, la pantalla o menú de incio, en la cual podemos ver el primer caso de "fallo" al que nos hemos preparado. Ya que de elegir una opción que no esté listada en el menú este arrojará un mensaje de invalidación y volverá a imprimir el menú. Este ciclo solo se rompe al elegir una opción real.

Seguido de ello, y yendo en orden descendiente, tenemos la opción de crear una sola comida a partir de los datos del usuario. Dicha comida consistirá en una elección balanceada y semi al azar de alimentos de diferentes grupos, los cuales cumplan con las necesidades calóricas del usuario. Esto lo hace a través de un método el cual checa cuantas comidas has ordenado ese día para después generarla directo de la clase "Menu Recomendado" y agregarla a tu historial. La

clase de menú recomendado recordemos que es donde convergen todas las líneas de pertenencia y herencia de las demás clases.

```
M
Fecha: 2023-06-16 20:26:14
Tu indice de masa es bajo, te recomendamos un consumo de altas calorias
        Alimento        Gramos  Calorias
        Salmon          100     206
        Avena           50      190
        Platano         50      48
        Brocoli         50      17
        Mantequilla             20      140
Calorias totales de comida sugerida: 601
```

Usando solamente la información de un usuario, genera un platillo que ayude a una dieta adaptada a la persona. Esto primero registrando la fecha y hora actual a través de la biblioteca “ctime”; imprimiendo dicha información y procediendo a cargar el catalogo desde al archivo txt (el cual está ordenado por grupos y dicho grupo en especifico por calorías descendentemente). Con “azar” nos referimos a lo antes mencionado del orden dentro del mismo txt, dentro del cual se extraen al azar uno de los primeros dos si se ocupan de altas calorías; de los siguientes dos si se ocupan medias, y de los finales dos en caso de bajas necesidades calóricas. Ya que se extraen solo dos por grupo alimenticio del catalogo y esos se imprimen.

Desde aquí podemos ir hasta el último tema que visto y ver el primer caso de fallo, control de excepciones. Pues al momento de leer un archivo externo es posible fallar debido a varias razones (entre las cuales la más importante es la corrupción, movimiento inesperado del archivo, o simplemente que este vació) en cuyo caso hemos preparado contramedidas de excepciones para indicar un fin del programa; en este caso indicando un error de abrir el archivo.

Hablando de ello, el funcionamiento detras del catalogo (siendo este con la librería “*fstream*”) le permite extraer seis elementos por grupos y pasar esos seis vectores a atributos de alimentos, como protocolo de recolección de datos. Después de esto, septima línea es el nombre

del grupo al que pertenecen los alimentos antes de la misma y cada octava está vacía. Así generando un catalogo dentro de un objeto de catalogo vacío y dando los atributos de cada alimento; y cada seis ciclos metiendo el grupo al catalogo y saltando una línea para empezar de nuevo. Esto hasta que el catalogo contenido en el documento de texto este completo.

```
D
Fecha: 2023-06-16 20:49:45
Tu indice de masa es bajo, te recomendamos un consumo de altas calorias
        Alimento        Gramos  Calorias
        Salmon          100     206
        Pan             100     150
        Pera            50      28
        Brocoli         50      17
        Mantequilla             20      140
Calorias totales de comida sugerida: 541
Fecha: 2023-06-16 20:49:45
Tu indice de masa es bajo, te recomendamos un consumo de altas calorias
        Alimento        Gramos  Calorias
        Salmon          100     206
        Pan             100     150
        Platano         50      48
        Zanahoria               50      20
        Mantequilla             20      140
Calorias totales de comida sugerida: 564
Fecha: 2023-06-16 20:49:45
Tu indice de masa es bajo, te recomendamos un consumo de altas calorias
        Alimento        Gramos  Calorias
        Salmon          100     206
        Avena           50      190
        Pera            50      28
        Zanahoria               50      20
        Helado          150     310
Calorias totales de comida sugerida: 754
Limite de comidas generadas alcanzado!
```

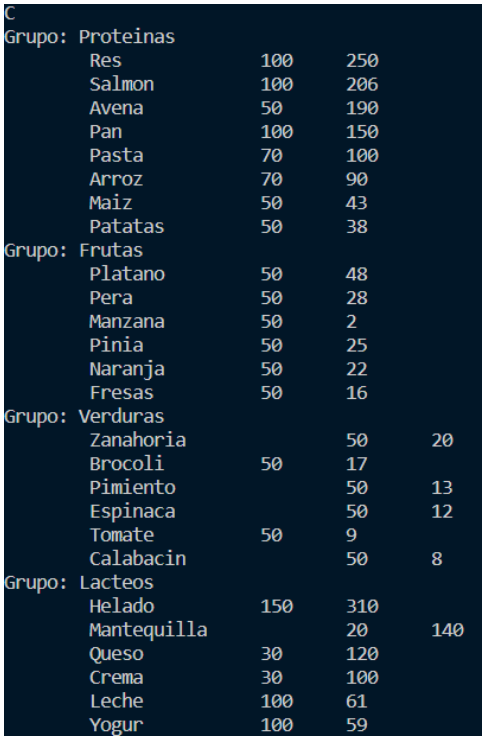
Después la siguiente opción es D de “Día”, esto nos permite generar el resto de comidas del día. Comparando en el historial cuantas hemos generado por día nos permite rellenar las posibles cuatro que se nos permite por día. Simplemente usa el mismo método antes mencionado para generar varias comidas en un menú, y en caso de haber generado comidas antes solo sigue el mismo protocolo las veces que te queden por día.

```
H
-------- Menus generados hoy --------
Fecha: 2023-06-16 20:49:42
        Salmon          100     206
        Avena           50      190
        Platano         50      48
        Brocoli         50      17
        Mantequilla             20      140
Fecha: 2023-06-16 20:49:45
        Salmon          100     206
        Pan             100     150
        Pera            50      28
        Brocoli         50      17
        Mantequilla             20      140
Fecha: 2023-06-16 20:49:45
        Salmon          100     206
        Pan             100     150
        Platano         50      48
        Zanahoria               50      20
        Mantequilla             20      140
Fecha: 2023-06-16 20:49:45
        Salmon          100     206
        Avena           50      190
        Pera            50      28
        Zanahoria               50      20
        Helado          150     310
-----------------------------------
```

Aquí podemos ver el uso de la función historial, la cual simplemente lee el historial antes mencionado de comidas generadas al día para que el usuario pueda revisar que cosas ha pedido en el transcurso del día.

```
I
--- INFO DEL USUARIO ---
Nombre: Jorge
Edad: 18
Estatura: 175 m
IMC: 0.00267755
Calorias consumidas hoy: 2460
```

En esta parte usamos herencia desde la clase persona a usuario para mover los atributos, además de sobrecarga de usuario para poder imprimir con un simple “cout” (cosa que hicimos en cada clase que tocará el main; entiéndase usuario, catalogo y menu recomendado). Además nos consigue sus calorías consumidas hoy según el historial generado por usuario, sacando y sumando las calorías directas de la variable caloria de cada alimento en sus comidas.



Finalmente, podemos ver el catalogo cargado desde el txt en expresión simple. Como dije, impreso con sobrecarga, y con una excepción de falla en caso de no haber cargado un catalogo aún, este nos marcara que está vacío. Usando los propios métodos de catalogo ya explicados nos genera una renderización de todos sus componentes en el orden y arreglo correcto. Con la cantidad a consumir en gramos como primer valor y sus calorías por porción como segundo.

Ya como opción final nos permite salir del programa con un *return 0* final.

Conclusión

Como conclusión final de proyecto podemos decir que no solo aplicamos todo lo visto en clase, sino que varias veces en otros trabajos y sobretodo en este proyecto nos tocó pensar fuera de la caja e investigar bastante para dar solución a problemas como lo fue la lectura de archivos o la generación de fechas y horas por reporte. Y saber temas como sobrecarga y excepciones nos

facilito mucho el trabajo ya que permite no solo programar más fácil sino controlar los métodos y sistemas de manera más sencilla.

Teniendo un *failsafe* en todo caso con los catch, y pudiendo usar una sola línea para lo que serían al menos diez de no ser por sobrecarga. Sumado a esto, el manejo de un sistema ideado en una fase de pre-producción del proyecto nos permitió usarlo como una especie de plano de construcción al momento de guiarnos con lo que queríamos hacer y como adaptarnos para alejarnos lo menos posible de lo propuesto inicialmente.

En verdad nunca se trató de una app de salud o un simple menú inteligente, se trató siempre de como integraríamos proceso complejos de una manera ordenada y eficiente para poder procesar información de manera eficiente y con estandares reales (entiendase laborales); en paquetes registrables y rastreables a través de nuestro propio sistema construido y el de la maquina al guardar la información.

Citas Bibliográficas

Para adultos. (2022, August 29). Centers for Disease Control and Prevention. https://www.cdc.gov/healthyweight/spanish/assessing/bmi/adult_bmi/index.html#:~:text=%C2%BFC%C3%B3mo%20se%20calcula%20el%20IMC%3F,-El%20IMC%20se&text=Con%20el%20sistema%20m%C3%A9trico%2C%20la,obtener%20la%20estatura%20en%20metros